TUTORIAL EASYBPMS

TECNOLOGIAS

1) jBPM: 6.2.0.Final
2) BPMN2Modeler: 1.0.4 Final (Kepler)

PRÉ-REQUISITO

Criar um projeto jBPM Maven **ou** então adicionar as dependências jBPM (http://docs.jboss.org/jbpm/v6.2/userguide/jBPMIntegration.html#d0e19126) ao pom do projeto

MODELAGEM DO PROCESSO

1) Propriedades do Processo
    a. **Process Id** – Nome da entidade de domínio que inicia o processo. Nomenclatura: nomedopacote_nomedaclasse
    b. **Interfaces** – Name (nome da classe que contém o serviço), Implementation (nomedopacote.nomedaclasse), Operation (nome do método da classe que será chamado para executar o serviço), In Message (mensagem aleatória e tipo aleatório)
    c. **DataItems** – Variáveis do processo que armazenarão valores durante a execução. Variáveis obrigatórias (id e tenancy). Nomenclatura: nomedopacote_nomedaclasse_nomedoatributo

2) Propriedades Service Task
    a. **Implementation** – Tipo de implementação do serviço. Ex.: Java
    b. **Operation** – Nome da Interface/Método

3) Propriedades Manual Task e Script Task
    a. Implementação de scripts em Java

4) Propriedades User Task
    a. **Actors** – Nome da Lane (Grupo de Usuário)

b. **InputParameters** – Parâmetros de entrada da atividade. Mapeamento LocalVariable (Property) –> Input Data (Parameter). Obs.: Todos parâmetros devem conter no nome a entidade de domínio que executa a tarefa. Obrigatórios: id (Int) - entidade que executa a tarefa, tenancy (String) – valor do tenancy no qual o usuário que executará a tarefa pertence. Nomenclatura: easybpms_nomedopacote_nomedaclasse_nomedoatributo

c. **OutputParameters** – Parâmetros de saída da atividade. Mapeamento OutputData (Parameter) -> LocalVariable (Property). Obs.: Se houver um gateway após essa atividade, o parâmetro de saída é necessário para que o gateway decida o caminho a seguir no fluxo

5) Gateways
   a. **Exclusive Gateway** – Escolhe uma atividade. Deve conter uma condição que retorna o valor da LocalVariable (Property) para cada caso a fim de definir o caminho do fluxo
   b. **Parallel Gateway** – Escolhe todas as atividades

GERAÇÃO DE CÓDIGO

6) Criar pacote na aplicação que receberá o Context (arquivo gerado automático com as definições modeladas no processo bpmn e o mapeamento Observable – Observer). Nomenclatura: Criar o source src/easybpms/java. Dentro do source, criar o pacote com.easybpms.codegen

JARS UTILIZADOS

7) Executar jar codegen
   a. **1ª caixa de diálogo** – caminho da pasta que contém os processos bpmn
   b. **2ª caixa de diálogo** – caminho do projeto que contém o source e o pacote criado na etapa anterior
8) Adicionar jars easybpms e jbpm ao classpath do projeto

CONFIGURAÇÃO BDs

9) Adicionar persistence.xml no src/main/resources/META-INF contendo as configurações de BD do easybpms e jbpm
10) Criar pacote/classe com.jbpm.bd.Connection e estender classe do easybpms AbstractConnection. No construtor definir as configurações de banco de dados do jbpm utilizando bitronix. Adicionar lib btm versão 2.1.4 ao classpath do projeto

CLASSES DE SERVIÇO

11) Adicionar pacote na aplicação com as classes de serviço, onde o nome da classe corresponde à interface criada na modelagem do processo, e o método corresponde à operação criada na modelagem de processo

USUÁRIOS E GRUPOS DE USUÁRIO

12) Criar classe GrupoUsuario e implementar a interface IUserGroup. Criar classe CRUDGrupoUsuario e no método create dessa classe chamar CRUDUserGroup.create do easybpms, passando como parâmetro o grupo de usuário criado na aplicação. Obs.: O nome do grupo de usuário corresponde à lane modelada no processo
13) Criar classe Usuario e implementar a interface IUser. Criar classe CRUDUsuario e no método create dessa classe chamar CRUDUser.create do easybpms, passando como parâmetro o usuário criado na aplicação. Obs.: idApp e name corresponde ao id e nome do usuário na aplicação respectivamente, tenancy corresponde ao valor do tenancy no qual o usuário pertence, userGroupNames corresponde a uma lista com os nomes dos grupos de usuário (getName() do GrupoUsuario) a qual o usuário pertence.

ENTIDADES DE DOMÍNIO

14) Criar classes de domínio que irão interagir com o processo de negócio. Para cada classe implementar a interface IDomainEntity do easybpms. Obs.: id corresponde ao id da entidade de domínio criada na aplicação e tenancy corresponde ao valor do tenancy no qual os usuários que pertencem a ele poderão ser escolhidos para executar a atividade

15) Para cada entidade de domínio da aplicação que irá interagir com o processo, implementar os atributos de acordo com as variáveis de processo (LocalVariable) criadas no momento da modelagem. Além disso, inicializar os atributos de tipo não primitivo com vazio. Ex.: String atributo = “”; Isso é necessário somente para os atributos que serão utilizados como variáveis de processo, pois eles serão mapeados para os parâmetros de entrada das atividades
16) Nas classes DAO das entidades de domínio que interagem com o processo, estender “CRUDObservable” do easybpms. A nomenclatura dessas classes deve ser CRUD + “nome da entidade”, pois são observáveis. Ex.: CRUDOcorrencia
17) Ao criar ou atualizar uma entidade de domínio, chamar o método “notifyObservers” passando como parâmetro a instância da entidade de domínio. Obs.: Os atributos id e tenancy da entidade que serão mapeados para os parâmetros de entrada das atividades devem ser inicializados ao criar a instância da entidade. Para executar cada atividade de usuário do processo deve ser feito um “notifyObservers”.